NAVIGATION CHIP – Project Description

In this Project there are around 42K flops. (106 scan chains each with 402 flops)

Functional Clock frequency: 400MHz

Scan Clock frequency: 25MHz

No. of Chains = 106

Length = 402 flops per chain

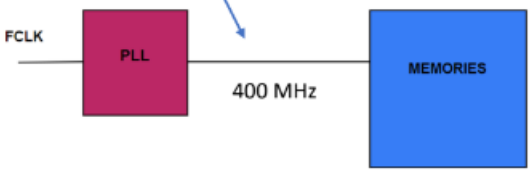
Compression Ratio = 53x

Tools:-

MBIST: Tessent MBIST
Scan Insertion: Tessent Scan
EDT: Tessent TestKompress
ATPG: Tessent TestKompress
Simulation: QuestaSim

DEFINING CLOCK

add_clocks 0 F_CLK -period 20 ns -pulse_always
add_clocks PLL/CLKOUT -REFERENCE F_CLK -FREQ_Multiplier 8 -label pll_out



F_CLK = 20 ns

Freq multiplier = 8
= 8x(1/20)
=400 MHz

IJTAG NETWORK

Top level port

set_attribute_value DEBUG_MODE -name function -value tck

set_attribute_value GPIO[7] -name function -value tdi

set_attribute_value SYS_MODE[2] -name function -value tms

set_attribute_value GPIO[10] -name function -value trst

set_attribute_value GPIO[6] -name function -value tdo

ANALYSIS> set_config_value DefaultsSpecification(user)/DftSpecification/use_rtl_cells off

Now, during process_dft_specification, library cells specified using dft_cell_selection will be used instead of their RTL equivalents.

- use_rtl_synchronizer_cell : on | off | auto ;

  A property that when set to "on", forces the tool to create and instantiate an RTL synchronizer cell

IJTAG NETWORK (contd…)

Command :

check_design_rules
set spec [create_dft_specification]

Create an initial DFT specification.

IJTAG NETWORK (contd…)

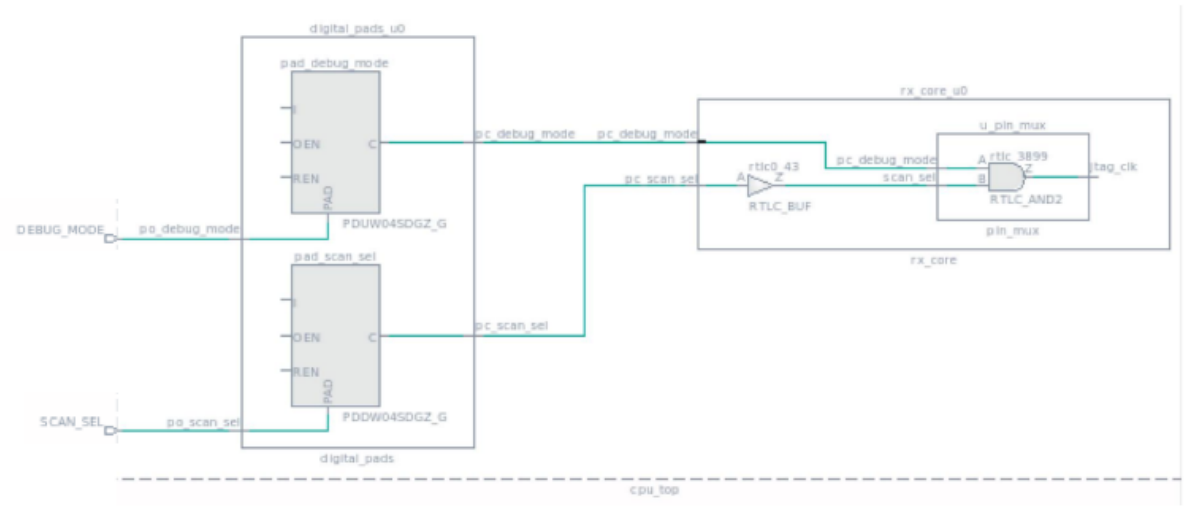
```
read_config_data -in_wrapper $spec/IjtagNetwork  -from_string {
   HostScanInterface(tap_internal) {
   Interface {
      tck : /rx_core_u0/u_pin_mux/jtag_clk;
      trst : /rx_core_u0/u_pin_mux/jtag_trstn;
      tms : /rx_core_u0/u_pin_mux/jtag_tms;
      tdi : /rx_core_u0/u_pin_mux/jtag_tdi;
      tdo : /rx_core_u0/u_pin_mux/jtag_tdo;
      tdo_en : /rx_core_u0/u_pin_mux/tdo_oen;
      tdo_en_polarity : active_low;
                        }
               }
}

move_config_element
${spec}/IjtagNetwork/HostScanInterface(tap)/Tap(main) -in_wrapper
${spec}/IjtagNetwork/HostScanInterface(tap_internal)
```

Consider this pin-mux output as tck

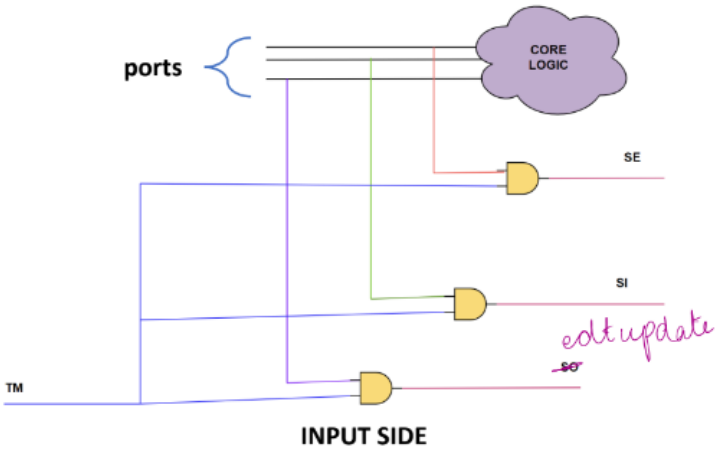
This command will tell the tool that the internal IJTAG declarations are related to the top level ports given in slide 3.

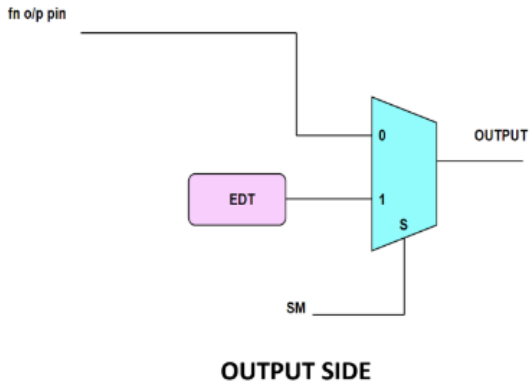
IJTAG NETWORK (contd…)



The above logic gets inserted when the commands in slides 4,5 and 6 are executed.

PIN-MUX LOGIC





To share DFT port with the functional ports. It is to reduce the package size.

PIN MUX LOGIC (contd…)

```
dict set ::auxiliary_data_dict SYS_MODE[1]
auxiliary_input_pin
rx_core_u0/u_pin_mux/scan_enable
```

→ Input pin

```
dict set ::auxiliary_data_dict SYS_MODE[1]
auxiliary_input_enable_pin
digital_pads_u0/pc_scan_sel
```

→ Control pin

EDT and OCC INSERTION

LEVEL 4 Project Script

```
set spec [create_dft_specification -sri_sib_list
{occ edt} ]

read_config_data -in $spec -from_string {
 OCC {
  ijtag_host_interface : Sib(occ);
    }
}
```

This is similar to the script which we have seen in Level 3 projects.
The only difference is in Level 3 projects script, the Controller and clock intercept node is mentioned. But in Level 4 project, it will be written using foreach loop (Next Slide)

LEVEL 3 Project Script

```
set spec [create_dft_spec -sri_sib_list {occ edt}]

read_config_data -in_wrapper $spec -from_string {
 Occ {
  ijtag_host_interface : Sib(occ);
  Controller(clka) {
   clock_intercept_node : clka;
  }
  Controller(clkb) {
   clock_intercept_node : clkb;
  }
  Controller(clkc) {
   clock_intercept_node : clkc;
  }
 }
```

EDT and OCC INSERTION (contd…)

```
set id_clk_list [list \
clk1 /rx_core_u0/clk_pll \
]
foreach {id clk} $id_clk_list {
  set occ [add_config_element OCC/Controller($id) -in $spec]
  set_config_value clock_intercept_node -in $occ $clk
}
report_config_data $spec
```

Controller(clk1)

```
Controller(clk1) {
     clock_intercept_node :
rx_core_u0/clk_pll ;
  }
```

Significance of using foreach loop:

If we have more no. of clocks in our design, we can just append it in our list. OCC will be automatically inserted at these points.

EDT and OCC INSERTION (contd…)

```
EDT {
  ijtag_host_interface : Sib(edt);
   Controller (c1) {
    longest_chain_range : 470, 475;
    scan_chain_count : 107;
    input_channel_count : 2;
    output_channel_count : 2;
     Connections +{
      EdtChannelsIn(1) {
      }
      EdtChannelsIn(2) {
      }
      EdtChannelsOut(1) {
      }
      EdtChannelsOut(2) {
      }
     }
   }
 }
```

Mentioning the parameters used in EDT configuration :

- longest_chain_range → min , max range of chain length
- scan_chain_count → number of internal scan chains
- input_channel_count → number of external scan input channels
- output_channel_count --> number of external scan output channels

EDT channels should have to be connected to the pin mux output.
Script for what has to be connected to the EDT Channels will be mentioned in the next slide.

EDT and OCC INSERTION (contd…)

```
set_config_value port_pin_name -in $spec/EDT/Controller(c1)/Connections/EdtChannelsIn(1)
[get_single_name [get_auxiliary_pins GPIO[15] -direction input]]
```

It will automatically take the pin mux output.

```
dict set ::auxiliary_data_dict GPIO[15] auxiliary_input_pin rx_core_u0/u_pin_mux/scan_in[0]
dict set ::auxiliary_data_dict GPIO[15] auxiliary_input_enable_pin digital_pads_u0/pc_scan_sel
```

PREPARING DFT LOGIC FOR SYNTHESIS

```
set preserve_instances [tessent_get_preserve_instances icl_extract]
set_boundary_optimization $preserve_instances false
```

Turning off boundary optimization for instances which can't be optimized.

```
set_app_var compile_enable_constant_propagation_with_no_boundary_opt false
```

If this command was to set to true, it will it will propagate constants across hierarchical boundaries.

It will not propagate constants across hierarchical boundaries.

PREPARING DFT LOGIC FOR SYNTHESIS (contd…)

set_app_var compile_seqmap_propagate_high_effort false

If this command was to set to true, the synthesis tool will remove the registers which can't escape from their reset state.

It will not remove registers which can't escape from their reset state.

set_app_var compile_delete_unloaded_sequential_cells false

If this command was to set to true, the synthesis tool will remove the flops whose output is not connected anywhere.

It will not remove the flops whose output is not connected anywhere.

PREPARING DFT LOGIC FOR SYNTHESIS (contd…)

set_size_only -all_instances [tessent_get_size_only_instances]
set_size_only [get_cells tessent_persistent_cell_* -hier -filter {is_hierarchical==false}] -all_instances

Synthesis tool can optimize only size and only for leaf cells (it can't optimize for modules)

set_ungroup

Sets the ungroup attribute on the specified objects so that they are ungrouped (collapsed to one design level) before they are optimized.

SCAN INSERTION

```
set edt_instance [get_name_list [get_instance -of_module \
                [get_name [get_icl_module -of_instances chip_top* \
                -filter  tessent_instrument_type==mentor::edt]]] ]
add_scan_mode edt_mode -edt_instance $edt_instance
analyze_scan_chains
report_scan_chains
insert_test_logic
exit
```

Snapshot from Mentor Graphics Manual

```
set edt_instance [get_name_list [get_instance -
of_module [get_name [get_icl_module -filter
tessent_instrument_type==mentor::edt]]]]


 add_scan_mode edt_mode -include_chain_families
{occ_chain core_chain} -single_cluster_chains on \
                                -edt_instance
$edt_instance   -port_index_start_value 1 \
                                -
port_scalar_index_modifier 1 -
single_clock_edge_chains off -
single_clock_domain_chains On
```

Script

SCAN INSERTION (contd…)

```
set edt_instance [get_name_list [get_instance -
of_module [get_name [get_icl_module -filter
tessent_instrument_type==mentor::edt]]]]


 add_scan_mode edt_mode -include_chain_families
{occ_chain core_chain} -single_cluster_chains on \
                                -edt_instance
$edt_instance   -port_index_start_value 1 \
                                -
port_scalar_index_modifier 1 -
single_clock_edge_chains off -
single_clock_domain_chains On
```

add_scan_mode edt_mode command to conn
the scan chains to the EDT signals and EDT
hardware that has been inserted during the EI
insertion.

An optional integer that specifies the starting
port index.

An optional switch and integer that
specifies the increment used between
each chain port indices. The default is
1.

Explanation in the next
slide

SCAN INSERTION (contd…)

SCAN INSERTION (contd…)

analyze_scan_chains	→	Distributes scan elements into new scan chains.
insert_test_logic	→	Inserts the test structures and stitches up the scan chain to increase the design's testability.